ProsumerFX: Mobile Design of Image Stylization Components

Tobias Dürschmid
Hasso Plattner Institute,
University of Potsdam, Germany

Maximilian Söchting
Hasso Plattner Institute,
University of Potsdam, Germany

Amir Semmo
Hasso Plattner Institute,
University of Potsdam, Germany

Matthias Trapp
Hasso Plattner Institute,
University of Potsdam, Germany

Jürgen Döllner
Hasso Plattner Institute,
University of Potsdam, Germany

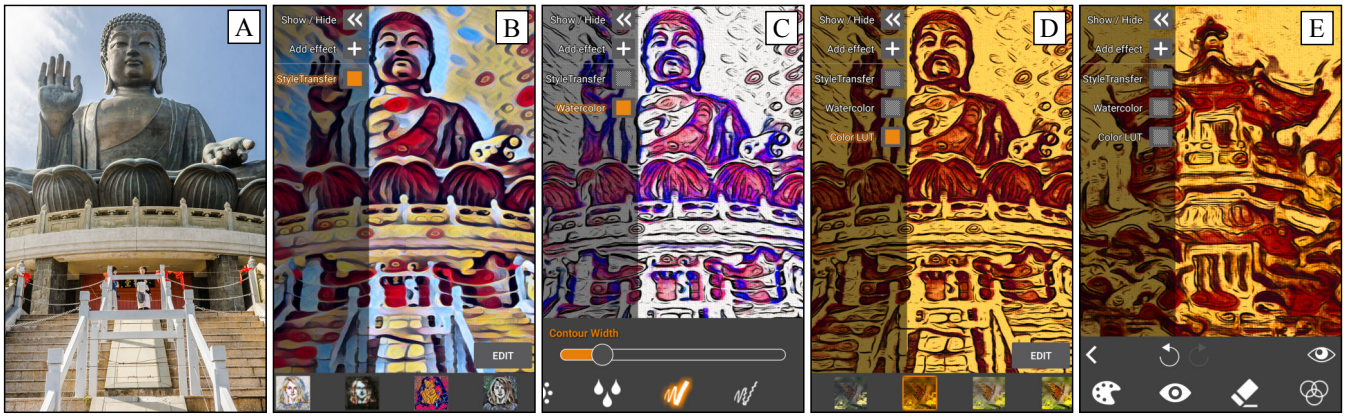

Figure 1: Use case of the presented app for creating and sharing new visual styles: (A) Original photo; (B) adding a neural style transfer effect with corresponding preset; (C) adding a watercolor filter with adjusted parameters; (D) adding a color filter with corresponding preset; (E) sharing the created parameterizable visual effect and using it on another device.

ABSTRACT

With the continuous advances of mobile graphics hardware, high-quality image stylization—e.g., based on image filtering, stroke-based rendering, and neural style transfer—is becoming feasible and increasingly used in casual creativity apps. The creative expression facilitated by these mobile apps, however, is typically limited with respect to the usage and application of pre-defined visual styles, which ultimately do not include their design and composition—an inherent requirement of prosumers. We present *ProsumerFX*, a GPU-based app that enables to interactively design parameterizable image stylization components on-device by reusing building blocks of image processing effects and pipelines. Furthermore, the presentation of the effects can be customized by modifying the icons, names, and order of parameters and presets. Thereby, the customized visual styles are defined as platform-independent effects and can be shared with other users via a web-based platform and database. Together with the presented mobile app, this system approach supports collaborative works for designing visual styles, including their rapid prototyping, A/B testing, publishing, and distribution. Thus, it satisfies the needs for creative expression of both professionals as well as the general public.

SA '17 Symposium on Mobile Graphics & Interactive Applications , November 27-30, 2017, Bangkok, Thailand

CCS CONCEPTS

• **Human-centered computing** → **Collaborative content creation**; • **Computing methodologies** → **Image manipulation**;

KEYWORDS

Image stylization, design, effect composition, mobile devices, interaction, rapid prototyping

ACM Reference Format:
Tobias Dürschmid, Maximilian Söchting, Amir Semmo, Matthias Trapp, and Jürgen Döllner. 2017. ProsumerFX: Mobile Design of Image Stylization Components. In *Proceedings of SA '17 Symposium on Mobile Graphics & Interactive Applications* . ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3132787.3139208

1 INTRODUCTION

Interactive image stylization enjoys a growing popularity in mobile expressive rendering [Dev 2013]. Prominent mobile apps (e.g., Instagram, Snapchat, or Prisma) attract millions of users each day—a popularity that may also be reasoned by the increased likelihood of stylized photos to be viewed and commented on [Bakhshi et al. 2015]. Typically, these apps provide stylization components with predefined parameter values to synthesize artistic renditions of user-generated contents, and thus are primarily directed towards "consuming" rather than "producing" custom parameterizable visual styles to facilitate creative expression.

Providing users with interactive tools for low-level and high-level parameterization of image stylization components is a desired

goal in three respects: (1) to facilitate creative expression at multiple levels of control—e.g., for interactive non-photorealistic rendering [Isenberg 2016; Semmo et al. 2016b]; (2) for the prosumption of image and video processing operations (IVOs), i.e., by combining aspects of consumption and production [Ritzer et al. 2012; Ritzer and Jurgenson 2010]; and (3) for the creation and manipulation of visual styles and their user interface (UI) to enable rapid prototyping of IVOs. In particular, these are important aspects of app development to shorten the time-to-market, facilitate user-centered design, and thus increase customer satisfaction. However, the development of a system that considers these aspects faces multiple technical challenges:

Interactivity: The modifications of the IVOs should be applied with interactive performance to provide immediate visual feedback, e.g., when adding or removing an effect, or reordering the effect pipeline. In this respect, however, mobile devices typically provide very constrained processing capabilities and memory [Capin et al. 2008].

Device Heterogeneity: Mobile graphics hardware and APIs often vary considerably. On the one hand, IVOs should be hardware optimized, but on the other hand, the IVOs need to be platform-independent to be used on different devices.

Technology Transparency: The adaption of image processing operations to new technology (e.g., Vulkan or newer OpenGL versions) and new hardware features, as well as APIs, must be as simple as possible.

Reusable Building Blocks: IVOs should be handled as modular units that can be referenced by others and reused for the design of new image stylization components.

In previous works [Dürschmid et al. 2017; Semmo et al. 2016b], we described a framework for interactive parameterization of image filtering operations on three levels of control: (1) convenience presets; (2) global parameters; and (3) local parameter adjustments using on-screen painting metaphors [Semmo et al. 2016a]. Based on that framework, this paper presents a system that enables its users to easily modify and recombine IVOs, i.e., to create and share new image stylization components that support these three interaction concepts. To summarize, the contributions of this paper are as follows: (1) a concept for mobile prosumption of image stylization components is proposed, which enables rapid prototyping for professionals and casual creativity for the general public; (2) a document format for platform-independent persistence of IVOs is provided that allows to exchange image stylization components across different devices and platforms; and (3) a web-based platform demonstrates how created IVOs can be shared among users, thus supporting collaboration.

2 RELATED WORK

This section describes the basic background of the prosumer culture and discusses previous work of interactive composition as well as design of image stylization components on desktop and mobile platforms.

2.1 Prosumer Culture

A prosumer is a person "who is both producer and consumer" [Ritzer et al. 2012]. The term traces back to Alvin Toffler in 1980 [Toffler 1980]. However, the 21st century gives rise to the prosumer, especially because of the Web 2.0, which popularizes user-generated content [Fuchs 2010; Ritzer et al. 2012]. Prominent examples are Wikipedia, Facebook, YouTube, and Flickr. Nowadays, consumers want to become prosumers, co-create value, and be creative on their own [Prahalad and Ramaswamy 2004].

2.2 Desktop Applications

Artistic stylization of images and video has been explored particularly on desktop systems for the task of non-photorealistic rendering (NPR) [Kyprianidis et al. 2013; Rosin and Collomosse 2013]. Applications using NPR techniques, however, primarily implement stylization techniques as invariable rendering pipelines of pre-defined parameter sets that cannot be individually designed or composed. First interactive tools such as Gratin [Vergne and Barla 2015] or ImagePlay[1] enable interactive effect specification and editing of such pipelines, whereas contemporary 3D modeling software (e.g., Autodesk® Maya and 3ds Max) traditionally supports the professional creation of rendering effects, but which cannot be shared and applied to images or video on mobile platforms.

2.3 Mobile Applications

In recent years, mobile expressive rendering [Dev 2013] has gained increasing interest to simulate popular media and effects such as cartoon [Fischer et al. 2008], watercolor [DiVerdi et al. 2013; Oh et al. 2012], and oil paint [Kang and Yoon 2015; Wexler and Dezeustre 2012] for casual creativity and image abstraction [Winnemöller 2013]. For instance, the popular image filtering app Instagram uses vignettes and color look-up tables [Selan 2004] to create retro looks through color transformations. Thereby, users are able to adjust global parameters, such as contrast and brightness. More complex stylization effects are provided by neural style transfer apps [Gatys et al. 2016; Johnson et al. 2016] such as Prisma, which simulate artistic styles of famous artists and epochs, but typically do not provide explicit parameterizations of style components or phenomena. Conversely, this can be achieved by mask-based parameter painting apps such as BeCasso [Pasewaldt et al. 2016; Semmo et al. 2016a] and PaintCan [Benedetti et al. 2014], where image filtering techniques that simulate watercolor, oil paint, and cartoon styles can be adjusted with a high, medium and low level-of-control [Isenberg 2016]. The combination of neural style transfer with post processing via image filtering can be done with Pictory [Semmo et al. 2017b].

Given these applications to reflect the state of the art in semi-automatic image and video stylization, however, mobile users are only able to consume effects. By contrast, this work seeks to enable the modification and platform-independent sharing of user-defined visual styles. To achieve this, a system approach is proposed for collaborative designing of visual styles and their application across different platforms by providing reusable IVOs. Sharing reusable aspects of IVOs has already been addressed in previous apps, but which are typically limited to sharing *parameter configurations* for *pre-defined* effects, e.g., the app Snapseed uses QR codes. By contrast, the proposed system and app provide a more generalized approach by also enabling to share *user-defined parameterizable effects*.

[1] http://imageplay.io/

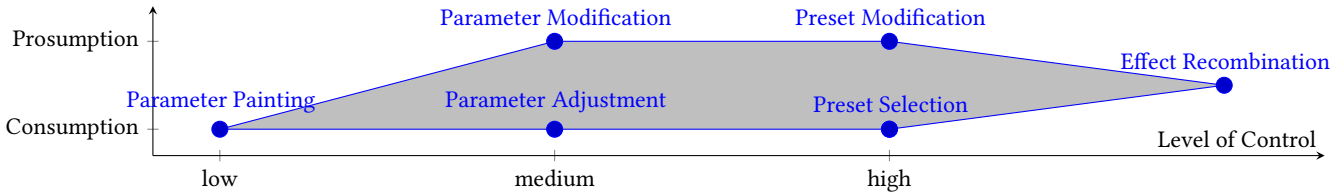

Figure 2: Creativity space with regard to level of control of the interactions. Low-level control is the adjustment of image details, e.g., local parameter painting. High-level control is the convenient modification of more abstract image properties, e.g., using presets or by adding / removing effects. The other dimension indicates whether the interaction just modifies the image by consuming an effect, or rather produces a new effect with a custom presentation.

3 DESIGNING EFFECTS ON MOBILE DEVICES

In contrast to existing image manipulation apps, the proposed approach enables users to modify image stylization components, save the new components, and share them online. These components consist of image and video processing operations (IVOs) (Sec. 3.1).

The client application covers two major IVO aspects: (1) the *processing function*, i.e., the visual transformation function applied to the input image to create an output image (Sec. 3.2), and (2) the *presentation*, i.e., the offered user experience (Sec. 3.3). The presented mobile app provides modifications of both aspects bundled in a single view to adjust both concurrently.

The proposed concept enables its users to parameterize IVOs on three levels of control [Isenberg 2016]: convenience presets, global parameters, and local parameter adjustments using on-screen painting [Semmo et al. 2016b]. Furthermore, prosumers can modify and recombine IVOs to create a new visual style that supports these three levels of control. A classification of the presented interaction concepts is given in Fig. 2.

3.1 Image and Video Processing Operations

To enable manipulation and combination of IVOs, a modular representation is required. To provide such a modular structure, hierarchical decomposition into effect pipelines, effects, and passes is used [Semmo et al. 2016b] and briefly described in the following.

Effect pipeline. An *effect pipeline* represents the highest-level abstraction of IVOs. It has a single input image, which can originate from different sources (e.g., camera or gallery), and produces a single output image to be displayed to the user. A pipeline mainly consists of an ordered list of effects that defines the processing steps of the component. By recombining effects, a variety of visual styles can be created.

Effect. An *effect* is a parameterizable IVO that receives a single image as input, and outputs one resulting image. It constitutes one atomic, sequenceable element of a visual style. Effects can delegate reoccurring processing steps (e.g., the computation of contours using a difference-of-Gaussians filter [Winnemöller et al. 2012]) to effect fragments. An *effect fragment* is a reusable part that can be shared among many effects. It can have multiple inputs and outputs and can delegate steps to other effect fragments. Effects and effect fragments can be parameterized by users via presets and parameters that map to technical inputs of rendering passes.

Rendering pass. A *rendering pass* is the lowest-level IVO. In the context of OpenGL ES, a rendering pass is basically a shader program having multiple inputs (textures or parameter values) and it usually renders one output texture or buffer. More complex passes (e.g., based on convolutional neural networks, ping-pong rendering, compute passes, or passes that execute Java code) can be used in combination with the standard render-to-texture passes.

3.2 Modifications of Processing Function

Modifications of the processing function (i.e., the visual transformation of an input image into an output image) of IVOs are the most crucial modifications, users can perform. Users get instant feedback, because these modifications influence the output image directly and immediately. Hence, this enables rapid prototyping of IVOs.

To extend the visual style, users can add an effect to the effects list. Subsequently, they can interactively explore the effect database of the web-based platform and choose one effect to add. The effect is downloaded, parsed, and appended to the current pipeline. To reduce the network traffic and to provide support for offline situations, the web pages and the downloaded effects are cached locally. Moreover, to remove an element of the visual style, effects can be removed from the current pipeline using a swipe gesture. To test new combinations, the execution order of the effects in the current pipeline can be changed by using a Drag & Drop interaction technique. The real-time rendering performance provides immediate feedback of the current effect order, even before releasing the currently dragged effect.

3.3 Modifications of the Presentation

An IVO can be presented in different forms tailored to different target groups, e.g., technical parameterization for experts or numerous convenient presets for novice users. The proposed app encourages the users' creativity by enabling them to adjust the user interface presentation of the IVO components. To provide a consistent set of convenient parameter configurations, users can add, remove, and reorder presets similar to the effect interactions described previously. Furthermore, users can remove and reorder parameters similar to presets. To customize the appearance of the parameters, presets or effect, users can rename them and change their representative images, e.g., icons, teaser images.

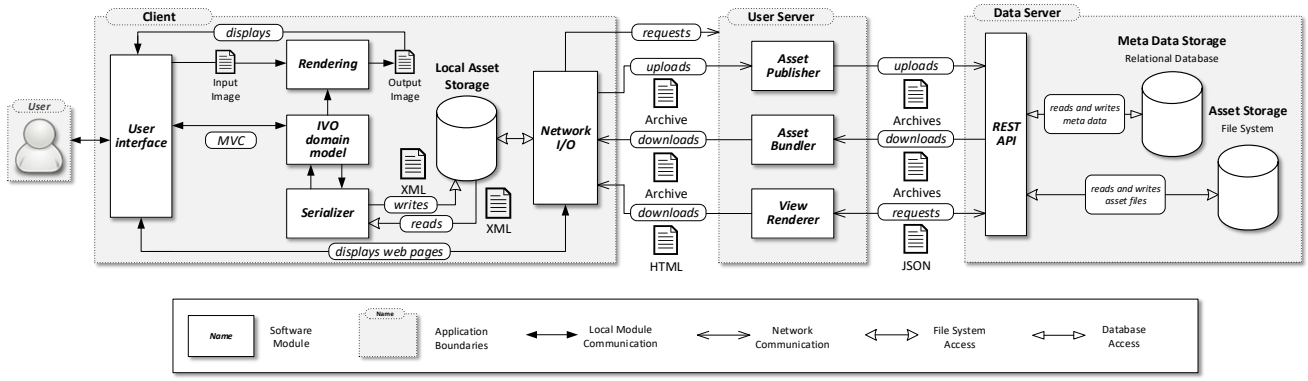

Figure 3: Overview of the system structure: Direct manipulation of image and video processing operations (IVOs) is provided by a Model-View-Controller (MVC) [Apple Inc. 2015]. The IVOs are persisted to extensible markup language (XML) files by a serializer [Riehle et al. 1997]. To exchange these effect XML files in conjunction with corresponding meta data, the user server offers a REST API. The data server stores this in form of *assets* (i.e, reusable effect modules), and the associated meta data.

4 IMPLEMENTATION

An overview of the proposed client-server system, which enables the design of visual effects on mobile devices, is given in Fig. 3. To handle the complexity of the IVOs implementations, a domain model is used, which represents the IVOs as classes and objects (Sec. 4.1). The direct manipulation of the domain model for IVOs is implemented using a Model-View-Controller [Apple Inc. 2015] (Sec. 4.2). To persist the state of the model, a serializer [Riehle et al. 1997] is used (Sec. 4.3). Based on this developed persistence concept, the server components implement the management and provisioning of IVO assets (Sec. 4.4) according to [Dürschmid 2017].

4.1 Domain Model of IVOs

The states and associations of IVOs can become very complex. Therefore, the IVOs described in Sec. 3.1 are implemented in corresponding classes and objects with their relationships (Fig. 4). Since effects and effect fragments share common responsibilities, such as maintaining passes, parameters as well as presets, a common super class is extracted. This object-oriented layer generalizes the graphics APIs by encapsulating calls to them in convenience methods. By abstracting a common interface for image processing, the domain model facilitates technology transparency.

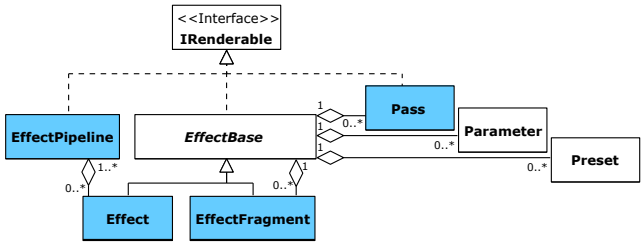

Figure 4: Overview of the architectural domain model of IVOs (blue) and related classes. Notation: UML 2.5 class diagram.

4.2 Direct Manipulation of IVOs

IVO changes performed by the user should directly influence the rendering state of the IVOs. Addressing usability, the user-perceived latency of modifications has to be interactive while keeping the memory consumption low. Furthermore, the persistence of the current state of the model needs to be as easy as possible.

To provide an architecture that supports all of these requirements, a Model-View-Controller (MVC) [Apple Inc. 2015] is used. Hence, the domain model has the role of the model that represents the domain logic of IVOs, stores the data and defines respective operations. The user interface has the role of the view that directs input events such as drag and drop gestures, or clicks on buttons to the controller that transforms the events to model changes. The controller calls the corresponding methods to manipulate the state of the model. Afterwards, the model notifies the views that registered of changes of the model (e.g., the name of a parameter) using the *Observer* design pattern [Gamma et al. 1995].

By offering information hiding [Parnas 1972], this separation provides domain model evolution. Since all information is kept in the model, the implementation of persistence is simplified. Removing an effect or preset immediately removes the underlying model. Thereby, memory consumption is minimized. The MVC provides interactive manipulation performance, because it immediately updates GPU resources.

4.3 Persistence of IVOs

Users should be able to share modified effects. Therefore, performed modifications should be durable in a platform-independent form. To reuse common building blocks (e.g., common algorithms, textures, or other resources) without duplication, the document format should support modularity.

IVOs are persisted using multiple XML files (Fig. 6) that define its content in a high-level, human readable form. To provide platform independence, each IVOs is separated in a platform-independent part (the *definition*) and possibly multiple platform-specific parts

(the *implementations*). This separation is designed to modularize the abstract, user-modifiable parts of an IVO in a single file. To support platform independence, the app keeps implementations transparent, i.e., users cannot modify implementations from within the app. Furthermore, to avoid duplication, reusable parts of implementations can be modularized in *effect fragments*.

The *effect definition* basically specifies the name of the IVO, its parameters (icon, name, type, and range), and its presets (icon, name, and values for each parameter). Furthermore, it can contain textures or geometry that define the visual appearance of the IVOs (e.g., canvas textures or color look-up tables) to be shared among all implementations. The *effect implementation* specifies the processing algorithm by defining rendering passes, corresponding shader programs, and a control flow that defines the execution order of the passes. An *implementation set* file defines the mapping of a definition to a corresponding implementation for each device specification.

4.4 Server-based Provisioning of IVOs

In order for effect authors to effectively share their work and profit from each others' creativity, a web-based platform with an effect database has been developed and set up. It enables users of the presented app and future apps to browse, download, and upload IVOs. In addition thereto, the server stores meta data such as metrics, example images, and textual descriptions of the IVO. Instead of simply uploading and manage complete IVOs, a modular *asset* format has been developed that reduces redundancy in uploaded effect files, optimizes bandwidth, and enables versioning of assets.

Asset Modularization. An asset can either contain an effect definition, an effect implementation, an implementation set, or an effect fragment in combination with corresponding resources, e.g., textures, icons, shaders, and feed-forward convolutional neural networks. To reduce duplication, reusable resources can be stored in separate, *common assets* and then be referenced (cf. Fig. 6). Since

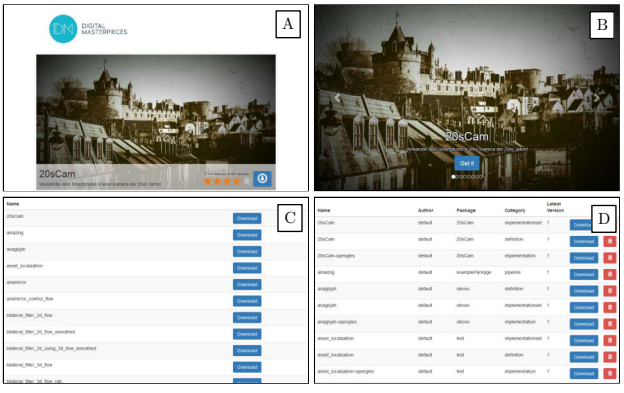

Figure 5: Four views of the web store interface: (A) a scrollable gallery; (B) a full screen carousel; (C) a basic list; (D) a detailed list. The two former views are based on *asset listings* and therefore include an example image and a short description for each asset. The latter two views display all available *assets* and are intended for development purposes only.

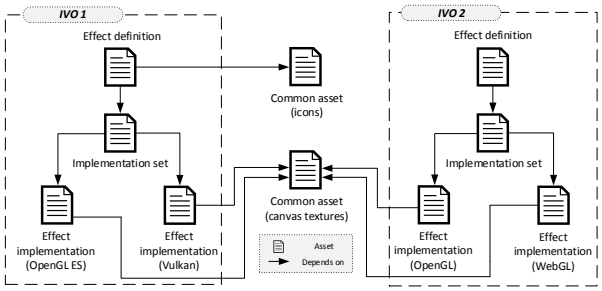

Figure 6: Two example IVOs, separated into appropriately categorized *assets*. IVO 1 is implemented for the two platforms OpenGL ES and Vulkan. The specific render pipelines are described in the respective effect implementation files. Due to the *asset* separation, all implementations of IVO 1 and IVO 2 can reference and reuse the same set of canvas textures.

assets are usually small sized, atomic units, setting up dependencies between different assets belonging to the same IVO is possible. These references are resolved by the *user server* once a client requests an asset with dependencies, delivering all depended assets using the *asset bundler module*. The separation into assets also enables versioning of atomic assets, i.e., users can submit updated versions of their assets and iteratively improve them. Older versions of assets can still be referenced and downloaded, but are not visible for consumption on the database web site. Since the device heterogeneity is addressed using implementation sets and implementations, the server can utilize this concept to deliver only compatible implementations. On requesting IVOs from the server, clients can submit a device specification describing their hardware specification and supported graphics APIs within the query. The server then resolves the optimal implementation that is compatible for the requesting device and delivers it as part of the IVO archive. This reduces required bandwidth and client storage space.

Server Architecture. In order to fulfill the three basic use cases (browsing, downloading, and uploading IVOs), a modularization over two different servers has been considered appropriate during the design stage (cf. Fig. 3).

The *Data Server* provides a REST (Representational State Transfer) API that allows for creation and retrieval of *asset* meta data, *asset* files, and user accounts. Authentication with a user account is required for the creation and deletion of *assets*. The data server REST API is designed for developers, as all responses are delivered as machine-readable JSON objects. The *Data Server* contains minimal domain knowledge as it knows of the properties of the *assets* but does not know any semantic meaning for any of the properties.

The *User Server* enables a more user-friendly interaction with the data provided by the *Data Server*. First, the *User Server* offers rendered HTML pages that allow for easy *asset* exploration through a web-store-like interface (cf. Fig. 5). Furthermore, the server allows users to request *asset bundles*, which are executable sets of *assets* put into one single archive [Söchting 2017].

5 USE CASES AND STYLIZATION EFFECTS

The proposed concept and implementation provides manifold use cases and a flexible design of stylization effects, which are outlined in the following.

5.1 Use Cases

Use cases and scenarios are directed to two main target groups: (1) novice users that utilize the system to share results of their casual creativity, and (2) technical artists that use the system for rapid prototyping and A/B testing of new image stylization components.

Consumption of IVOs. Components of IVOs can be consumed and applied to user-generated contents by the users of both target groups. In addition thereto, the individual parameters and presets of the respective IVOs can be edited and stored locally on device for reuse. This functionality represents also the basis for both target groups to create variants of IVOs. Furthermore, it can be coupled with notification or subscription models to keep users up-to-date.

Production of IVOs. By manipulating the processing function and the presentation of IVOs, users can produce a new IVO. This use case is common for both target groups. Rapid prototyping enables easy and quick creation of IVOs. In the context of casual creativity, this is the foundation to provide a tool that can be used by novice users. Technical artists can apply the rapid prototyping features to quickly modify the presentation of IVOs between A/B testing sessions. Therefore, they can simply conduct multiple iterations and instantly respond to collected feedback by adjusting the parameter order, exchanging icons, or using more descriptive names. Afterwards, the created visual style can be integrated in a tailored stand-alone app, using a product line approach [Dürschmid et al. 2017].

Community as Channel for Prosumers. The web-based platform offers possibilities of downloading (*consuming*) and uploading (*producing*) IVOs. Therefore, it provides a channel for prosumers of IVOs to share their created components to different users and groups. Furthermore, they can rate IVOs and share them using linking or transferred using push notifications. This use case supports casual creativity by proving an open social media channel that enables sharing of the created results. Technical artists can use the community to distribute their work and to sell it to a wide range of customers.

5.2 Stylization Effects

By offering a modular document format that supports to reference common effects, effect fragments, shaders, or textures, the concept supports the creation and usage of re-usable building blocks. Thereby, two main objectives are of particular interest: the implementation of state-of-the-art stylization effects that were initially dedicated to desktop platforms; and the convenient combination of popular effects or buildings blocks within a single effect pipeline. The following examples are described for the paradigms of image filtering and example-based rendering [Kyprianidis et al. 2013].

Recreating state-of-the-art effects. We used generalized building blocks, such as bilateral filtering and flow-based smoothing for color abstraction as well as difference-of-Gaussians filtering for edge enhancement, together with effect-specific fragments to simulate popular media and styles. In particular, this comprises a Cartoon effect that additionally uses color quantization [Winnemöller et al. 2006]; oil paint filtering using an a specialized paint texture synthesis [Semmo et al. 2016c]; watercolor rendering using a distinguished wet-in-wet pass as described in [Wang et al. 2014] and a composition pass that blends multiple texture assets to simulate phenomena such as pigment dispersion [Bousseau et al. 2006]; and a pencil hatching effect that uses orientation information to align tonal art maps [Praun et al. 2001].

Designing new effect compositions. We used our system to combine each of the previously described filtering effects with a neural style transfer (NST, e.g., known from *Prisma*) and color transformation (e.g., known from *Instagram*), as shown in Fig. 1. This combined approach enables to transform images into high-quality artistic renditions on a global and local scale as proposed in [Semmo et al. 2017a]. Thereby, shading-based implementations of Johnson et al.'s feed-forward NST [Johnson et al. 2016] are used in a first processing stage to yield an intermediate result. Subsequently, joint bilateral upsampling [Kopf et al. 2007] of a low-resolution NST is used with the original input image to filter high-resolution images and maintain interactivity. The result is then processed using mentioned image filtering effects—e.g., watercolor rendering or oil paint filtering—to reduce fine-scale visual noise and locally inject characteristics of artistic media, which may be interactively refined by a user. Finally, color moods may be adjusted by using fast color transfer functions based on color look-up tables [Selan 2004].

6 RESULTS AND DISCUSSION

The effects used for measurements in this section are: *Toon* (bilateral filtering, local luminance thresholding for color quantization, and artistic contours, comprising 14 render-to-texture passes, five half-float textures and 16 byte textures); *Oilpaint* (flow-based painting strokes painted on a canvas texture, consisting of 18 render-to-texture passes, 9 half-float textures, and 16 byte textures); *Pencil Hatching* (flow-oriented example-based hatching with contours, consisting of 14 render-to-texture passes, five half-float textures, 17 byte textures, and 18 small tonal art map textures); *Watercolor* (simulated wobbling, edge darkening, pigment density variation, and wet-in-wet, comprising 22 render-to-texture passes, 13 half-float textures, and 25 byte textures); and *Color Transformation* (adjusting saturation, brightness, expose, contrast, gamma, lights, and shadows of the image, consisting of only one render-to-texture pass and one byte texture). Example images of the complex effects are shown in Fig. 7.

The used devices are the *Sony Experia Z3* (a medium-class smartphone with a Qualcomm Adreno 330 GPU (578 MHz), a 2.5 GHz

Figure 7: Overview of the effects used for evaluating the approach.

Effect	Z3 (HD)	S7 (HD)	S7 (FHD)	S7 (WQHD)	Pixel C (WQHD)
Pencil Hatching	14.2 fps	30.0 fps	14.8 fps	7.8 fps	9.8 fps
Oil paint	4.3 fps	11.8 fps	3.9 fps	1.6 fps	1.1 fps
Toon	13.1 fps	30.0 fps	15.0 fps	7.8 fps	7.8 fps
Watercolor	7.3 fps	21.6 fps	8.3 fps	4.2 fps	3.6 fps
Color Transform	29.8 fps	30.0 fps	30.0 fps	30.0 fps	30.0 fps

Table 1: Frame rate of the filter effects on different devices.

Effect	Z3	S7	Pixel C
Pencil Hatching	1.68 s	0.87 s	0.45 s
Oil paint	1.33 s	0.41 s	0.88 s
Toon	1.09 s	1.97 s	0.42 s

Table 2: Time for parsing effects.

quad-core Krait 400 CPU, and an HD 720 × 1280 display); the *Samsung Gallaxy S7* (a high-end smartphone with an ARM Mali-T880 MP12 GPU (650 MHz), an Octa-core (4×2.3 GHz Mongoose & 4×1.6 GHz Cortex-A53) CPU, and a Wide Quad High Definition (WQHD) 2560 × 1440 display that can also be changed to Full High Definition (FHD) and High Definition (HD); and the *Google Pixel C* (a high-end tablet with a Nvidia Maxwell GPU (850 MHz), a 1.9 GHz quad-core "big.LITTLE" ARMv8-A CPU, and a WQHD 2560 × 1800 display).

Interactivity. The presented app enables its users to modify high-quality effects during run-time. It immediately updates the rendering pipeline and displays the resulting image with interactive frame rates. Recombining or removing effects takes less than a millisecond, even with more than ten effects in the pipeline. Downloading, extracting, parsing, and adding an effect to the pipeline using a 100 MBit/s connection takes 1 s to 3 s in total. About half of this duration is utilized for parsing the effect files (cf. Table 2).

However, the time to display the result image depends on the performance of the IVOs and their parameterization. Complex artistic effects such as pencil hatching achieve a frame rate of 10 fps to 20 fps in FHD on the Samsung S7 and similar devices (cf. Table 1). Simpler effects achieve higher frame rates. The color transformation effect even reaches the devices' limit of 30 fps. However, neural style transfer targeting an output resolution of 1024^2 pixels computes 3 s to 15 s, depending on the device. To improve the rendering performance, the system re-renders only the passes and effects that have their inputs changed. Additionally, the app automatically reduces the preview quality while painting as well as in the camera live view.

Furthermore, the amount of effects is limited with respect to their memory consumption. If the rendering has a target resolution of WQHD, the S7 is able to hold nine pencil hatching effect in a pipeline. With a resolution of FHD, the limit grows to 12 pencil effects. With HD, even 14 pencil hatching effects can be rendered in one pipeline. Furthermore, the complexity of the effects influences the limit of effects in the pipeline. Six oil paint effects or six watercolor effects can be hold in one pipeline targeting WQHD on the S7. In contrast to this, more than 100 of the small color transformation effects can be rendered together.

Device Heterogeneity. The platform-independent document format and the separation of IVOs in definition and implementation addresses the device heterogeneity of the created effects. However, if one IVO supports only one target API, e.g., one pass uses OpenGL ES 3.1 features and does not provide a fall back for OpenGL ES 2.0, all components that contain it cannot be executed or manipulated on such a device. Hence, the concept enables platform independence but does not solve it completely on its own.

Technology Transparency. The proposed concept provides support for technology transparency of IVOs by defining a document format that abstracts concrete technologies. Therefore, adjustments to new rendering APIs do not affect existing assets. However, if a change to the document structure (e.g., an XML schema update) is required, it can easily be performed by the data server using migration scripts.

Reusable Building Blocks. The presented framework and XML-based document format has been successfully used for teaching image processing techniques with OpenGL ES in undergraduate and graduate courses. In total, 190 effects have been designed and developed using the presented document format. Since the students were able to create new high-quality IVOs, the document format can be considered simple enough for the use by less experienced developers.

7 CONCLUSIONS AND FUTURE WORK

We presented a system that enables novice users and professionals to create and manipulate their individual platform-independent, parameterizable image stylization components based on existing components, customize their presentation, as well as share them in a web-based community. This kind of prosumer culture goes beyond user-generated content, because users generate tools that enable other users to generate content. Our scenarios represent examples on how this concept provides a higher level of creativity for users and supports rapid prototyping and rapid A/B testing of visual styles for professionals. The presented concept is not limited to support mobile devices only, and can be used as blueprint for other implementation platforms such as desktop PCs or server-based rendering approaches.

7.1 Future Work

Low Level Effect Modifications. To extend the control that users have over their effects, the app might support interactions for low level effect modifications, especially on tablets. Possible extensions could be: exchanging processing steps of effect (e.g., choose another contour extraction algorithm) to optimize visual quality or performance; exchanging textures (e.g., canvas textures or color lookup tables) to adjust the visual attributes of the effect's processing function; or collapsing similar parameters to a single one to simplify the presentation of the effect.

Hinting Rendering Performance. When a user combines many effects that have low rendering performance, the total rendering time of the style might increase substantially. To give users advice on their created effects, the system might hint slow effect in the pipeline and give the user control of the rendering quality. Thereby, users could tailor the quality/performance ratio to their purpose.

ACKNOWLEDGMENTS

We thank Erik Griese, Moritz Hilscher, Alexander Riese, and Hendrik Tjabben for their contributions to the design and implementation of the presented system. This work was partly funded by the Federal Ministry of Education and Research (BMBF), Germany, for the AVA project 01IS15041B and within the InnoProfile Transfer research group "4D*n*D-Vis" (www.4dndvis.de).

REFERENCES

Apple Inc. 2015. Model-View-Controller. (21 Oct. 2015). https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

Saeideh Bakhshi, David A. Shamma, Lyndon Kennedy, and Eric Gilbert. 2015. Why We Filter Our Photos and How It Impacts Engagement. In *Proc. ICWSM*. AAAI Press, 12–21. http://www.aaai.org/ocs/index.php/ICWSM/ICWSM15/paper/view/10573/10484

Luca Benedetti, Holger Winnemöller, Massimiliano Corsini, and Roberto Scopigno. 2014. Painting with Bob: Assisted Creativity for Novices. In *Proc. ACM Symposium on User Interface Software and Technology*. ACM, New York, 419–428. https://doi.org/10.1145/2642918.2647415

Adrien Bousseau, Matt Kaplan, Joëlle Thollot, and François X. Sillion. 2006. Interactive Watercolor Rendering with Temporal Coherence and Abstraction. In *Proc. NPAR*. ACM, New York, 141–149. https://doi.org/10.1145/1124728.1124751

Tolga Capin, Kari Pulli, and Tomas Akenine-Möller. 2008. The State of the Art in Mobile Graphics Research. *IEEE Computer Graphics and Applications* 28, 4 (2008), 74–84. https://doi.org/10.1109/MCG.2008.83

Kapil Dev. 2013. Mobile Expressive Renderings: The State of the Art. *IEEE Computer Graphics and Applications* 33, 3 (2013), 22–31. https://doi.org/10.1109/MCG.2013.20

Stephen DiVerdi, Aravind Krishnaswamy, Radomir Mäch, and Daichi Ito. 2013. Painting with Polygons: A Procedural Watercolor Engine. *IEEE Transactions on Visualization and Computer Graphics* 19, 5 (2013), 723–735. https://doi.org/10.1109/TVCG.2012.295

Tobias Dürschmid. 2017. A Framework for Editing and Execution of Image and Video Processing Techniques on Mobile Devices. (26 July 2017). https://doi.org/10.13140/RG.2.2.13252.32648

Tobias Dürschmid, Matthias Trapp, and Jürgen Döllner. 2017. Towards Architectural Styles for Android App Software Product Lines. In *Proc. International Conference on Mobile Software Engineering and Systems*. IEEE Press, Piscataway, NJ, USA, 58–62. https://doi.org/10.1109/MOBILESoft.2017.12

Jan Fischer, Michael Haller, and Bruce H Thomas. 2008. Stylized Depiction in Mixed Reality. *International Journal of Virtual Reality* 7, 4 (Dec. 2008), 71–79. http://mi-lab.org/files/publications2008/fischer2008-ijvr.pdf

Christian Fuchs. 2010. Web 2.0, Prosumption, and Surveillance. *Surveillance & Society* 8, 3 (2 Sept. 2010), 288–309. https://ojs.library.queensu.ca/index.php/surveillance-and-society/article/view/4165

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1995. *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2016. Image Style Transfer Using Convolutional Neural Networks. In *Proc. CVPR*. IEEE Computer Society, Los Alamitos, 2414–2423. https://doi.org/10.1109/CVPR.2016.265

Tobias Isenberg. 2016. Interactive NPAR: What Type of Tools Should We Create?. In *Proc. NPAR*. Eurographics Association, Goslar, Germany, 89–96. https://doi.org/10.2312/exp.20161067

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *Proc. ECCV*. Springer International, Cham, Switzerland, 694–711. https://doi.org/10.1007/978-3-319-46475-6_43

Dongwann Kang and Kyunghyun Yoon. 2015. Interactive Painterly Rendering for Mobile Devices. In *Proc. International Conference on Entertainment Computing*. Springer International Publishing, Cham, Switzerland, 445–450. https://doi.org/10.1007/978-3-319-24589-8_38

Johannes Kopf, Michael F. Cohen, Dani Lischinski, and Matt Uyttendaele. 2007. Joint Bilateral Upsampling. *ACM Transactions on Graphics* 26, 3 (July 2007). https://doi.org/10.1145/1276377.1276497

Jan Eric Kyprianidis, John Collomosse, Tinghuai Wang, and Tobias Isenberg. 2013. State of the "Art": A Taxonomy of Artistic Stylization Techniques for Images and Video. *IEEE Transactions on Visualization and Computer Graphics* 19, 5 (May 2013), 866–885. https://doi.org/10.1109/TVCG.2012.160

Junkyu Oh, SeungRol Maeng, and Jinho Park. 2012. Efficient Watercolor Painting on Mobile Devices. *International Journal of Contents* 8, 4 (2012), 36–41. https://doi.org/10.5392/IJoC.2012.8.4.036

David Lorge Parnas. 1972. On the Criteria to Be Used in Decomposing Systems into Modules. *Commun. ACM* 15, 12 (Dec. 1972), 1053–1058. https://doi.org/10.1145/361598.361623

Sebastian Pasewaldt, Amir Semmo, Jürgen Döllner, and Frank Schlegel. 2016. BeCasso: Artistic Image Processing and Editing on Mobile Devices. In *Proc. SIGGRAPH ASIA Mobile Graphics and Interactive Applications*. ACM, New York, 14:1–14:1. https://doi.org/10.1145/2999508.2999518

Coimbatore Krishna Prahalad and Venkat Ramaswamy. 2004. *The future of competition: Co-creating unique value with customers*. Harvard Business Press.

Emil Praun, Hughes Hoppe, Matthew Webb, and Adam Finkelstein. 2001. Real-Time Hatching. In *Proc. SIGGRAPH*. ACM, New York, 581–586. https://doi.org/10.1145/383259.383328

Dirk Riehle, Wolf Siberski, Dirk Bäumer, Daniel Megert, and Heinz Züllighoven. 1997. Pattern Languages of Program Design 3. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, Chapter Serializer, 293–312.

George Ritzer, Paul Dean, and Nathan Jurgenson. 2012. The Coming of Age of the Prosumer. *American Behavioral Scientist* 56, 4 (2012), 379–398. https://doi.org/10.1177/0002764211429368

George Ritzer and Nathan Jurgenson. 2010. Production, Consumption, Prosumption. *Journal of Consumer Culture* 10, 1 (2010), 13–36. https://doi.org/10.1177/1469540509354673

Paul Rosin and John Collomosse (Eds.). 2013. *Image and Video based Artistic Stylisation*. Computational Imaging and Vision, Vol. 42. Springer, London/Heidelberg. https://doi.org/10.1007/978-1-4471-4519-6

Jeremy Selan. 2004. Using Lookup Tables to Accelerate Color Transformations. In *GPU Gems*. Addison-Wesley, 381–392. http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter24.html

Amir Semmo, Jürgen Döllner, and Frank Schlegel. 2016a. BeCasso: Image Stylization by Interactive Oil Paint Filtering on Mobile Devices. In *ACM SIGGRAPH 2016 Appy Hour*. ACM, New York, 6:1–6:1. https://doi.org/10.1145/2936744.2936750

Amir Semmo, Tobias Dürschmid, Matthias Trapp, Mandy Klingbeil, Jürgen Döllner, and Sebastian Pasewaldt. 2016b. Interactive Image Filtering with Multiple Levels-of-control on Mobile Devices. In *Proc. SIGGRAPH ASIA Mobile Graphics and Interactive Applications*. ACM, New York, 2:1–2:8. https://doi.org/10.1145/2999508.2999521

Amir Semmo, Tobias Isenberg, and Jürgen Döllner. 2017a. Neural Style Transfer: A Paradigm Shift for Image-based Artistic Rendering?. In *Proc. International Symposium on Non-Photorealistic Animation and Rendering*. CM, 5:1–5:13.

Amir Semmo, Daniel Limberger, Jan Eric Kyprianidis, and Jürgen Döllner. 2016c. Image Stylization by Interactive Oil Paint Filtering. *Computers & Graphics* 55, C (April 2016), 157–171. https://doi.org/10.1016/j.cag.2015.12.001

Amir Semmo, Matthias Trapp, Jürgen Döllner, and Mandy Klingbeil. 2017b. Pictory: Combining Neural Style Transfer and Image Filtering. In *ACM SIGGRAPH 2017 Appy Hour (SIGGRAPH '17)*. ACM. https://doi.org/10.1145/3098900.3098906

Maximilian Söchting. 2017. Design, Implementation and Web-based Provisioning of a Database for Image Processing Operations. (26 July 2017). https://doi.org/10.13140/RG.2.2.23550.48964

Alvin Toffler. 1980. *The Third Wave*. Bantam Books.

Romain Vergne and Pascal Barla. 2015. Designing Gratin, A GPU-Tailored Node-Based System. *Journal of Computer Graphics Techniques (JCGT)* 4, 4 (Nov. 2015), 54–71. http://jcgt.org/published/0004/04/03/

Miaoyi Wang, Bin Wang, Yun Fei, Kanglai Qian, Wenping Wang, Jiating Chen, and Jun-Hai Yong. 2014. Towards Photo Watercolorization with Artistic Verisimilitude. *IEEE Transactions on Visualization and Computer Graphics* 20, 10 (Feb. 2014), 1451–1460. https://doi.org/10.1109/TVCG.2014.2303984

Daniel Wexler and Gilles Dezeustre. 2012. Intelligent Brush Strokes. In *Proc. ACM SIGGRAPH Talks*. ACM, New York, NY, USA, 50:1–50:1. https://doi.org/10.1145/2343045.2343112

Holger Winnemöller. 2013. NPR in the Wild. In *Image and Video based Artistic Stylisation*, Paul Rosin and John Collomosse (Eds.). Computational Imaging and Vision, Vol. 42. Springer, London/Heidelberg, Chapter 17, 353–374. https://doi.org/10.1007/978-1-4471-4519-6_17

Holger Winnemöller, Jan Eric Kyprianidis, and Sven Olsen. 2012. XDoG: An eXtended Difference-of-Gaussians Compendium including Advanced Image Stylization. *Computers & Graphics* 36, 6 (Oct. 2012), 740–753. https://doi.org/10.1016/j.cag.2012.03.004

Holger Winnemöller, Sven C. Olsen, and Bruce Gooch. 2006. Real-Time Video Abstraction. *ACM Transactions on Graphics* 25, 3 (July 2006), 1221–1226. https://doi.org/10.1145/1141911.1142018